PETUNJUK PRAKTIKUM
DASAR SISTEM KOMPUTER

Disusun oleh :

Ahmad Azhari, S.Kom., M.Eng.

Ali Tarmuji, S.T., M.Cs.

Taufik Ismail, S.T., M.Cs.

Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Ahmad Dahlan

2018

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya, sehingga dapat diselesaikannya Petunjuk Praktikum ini. Sholawat dan salam semoga terlimpah kepada baginda Nabi Muhammad saw beserta keluarga, sahabat dan para pengikutnya.

Petunjuk Praktikum Dasar Sistem Komputer ini disusun dengan harapan dapat memberikan kemudahan bagi mahasiswa untuk memahami dasar-dasar sistem komputer terutama dalam pemrograman Bahasa rakitan (*assembly*). Berbagai contoh aplikasi pemrograman *assembly* diberikan yang meliputi debug, aritmatika, word process, dan penggunaan emu8086. Ucapkan terima kasih yang sebesar-besarnya diberikan kepada pihak semua pihak yang telah membantu dalam proses penyusunan petunjuk praktikum ini.

Akhirnya, tiada gading yang tak retak, kritik dan saran membangun kami harapkan untuk lebih sempurnanya petunjuk praktikum ini. Semoga ilmu yang diperoleh dari membaca dan mempraktekkan petunjuk praktikum ini dapat bermanfaat dan barokah.

Yogyakarta, Agustus 2018

Wassalam

Penulis

MODUL PRAKTIKUM

DASAR SISTEM KOMPUTER

Tim Dasar Sistem Komputer

DAFTAR ISI

PRAKTIKUM 01 : RANDOM ACCESS MEMORY

TUJUAN

1. Mahasiswa mampu menjelaskan tentang RAM.
2. Mahasiswa mampu menjelaskan kegunaan RAM.
3. Mahasiswa mampu menjelaskan penerapan RAM pada program komputer.

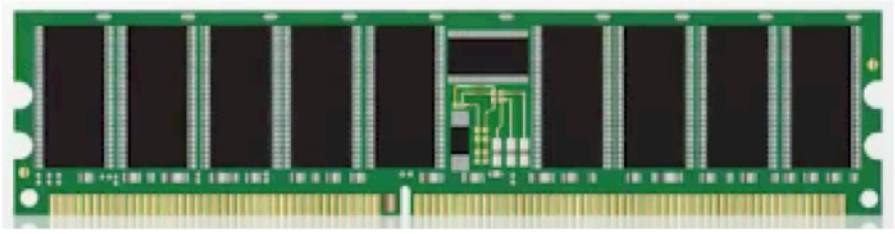
DASAR TEORI

RAM (*Random Access Memory*) adalah jenis penyimpanan data komputer. Perangkat RAM memungkinkan untuk mengakses data dalam urutan acak, yang membuatnya sangat cepat untuk menemukan sepotong informasi tertentu. Jenis penyimpanan tertentu lainnya bukan akses acak. Misalnya, hard disk drive dan CD akan membaca dan menulis data dalam urutan yang telah ditentukan. Rancangan mekanis dari perangkat ini menentukan bahwa akses data secara berurutan. Ini berarti bahwa waktu yang diperlukan untuk menemukan bagian informasi tertentu dapat sangat bervariasi tergantung dimana letaknya pada disk.

RAM (*Random Access Memory*) digunakan dalam sistem komputer sebagai memori utama. RAM dianggap memori *volatile*, yang berarti bahwa informasi yang disimpan hilang ketika tidak ada daya. Jadi, RAM digunakan oleh central processing unit (CPU) ketika komputer berjalan untuk menyimpan informasi yang perlu digunakan dengan sangat cepat, tetapi tidak menyimpan informasi apa pun secara permanen.

Perangkat RAM saat ini menggunakan sirkuit terpadu untuk menyimpan informasi. Ini adalah bentuk penyimpanan yang relatif mahal dan biaya per unit penyimpanan jauh lebih tinggi daripada perangkat seperti hard drive. Namun, waktu untuk mengakses data jauh lebih cepat untuk RAM yang kecepatannya melebihi biaya. Oleh karena itu, Komputer menggunakan sejumlah RAM untuk akses cepat, penyimpanan sementara informasi dan jumlah penyimpanan massal permanen yang tidak acak, seperti hard disk drive. Sebagai contoh, sistem komputer yang khas mungkin memiliki dua hingga delapan GB (gigabyte) RAM, sementara kapasitas penyimpanan hard disk drive bisa beberapa ratus GB atau bahkan satu TB (terabyte).

RAM memungkinkan data untuk dibaca atau ditulis dalam jumlah waktu yang hampir sama terlepas dari lokasi fisik data di dalam memori. Beberapa jenis RAM, yang memiliki Kecepatan, konsumsi daya dan teknologi yang berbeda. Gambar 1 menunjukkan contoh dari RAM pada komputer.



Gambar 1. *Random Access Memory* (RAM)

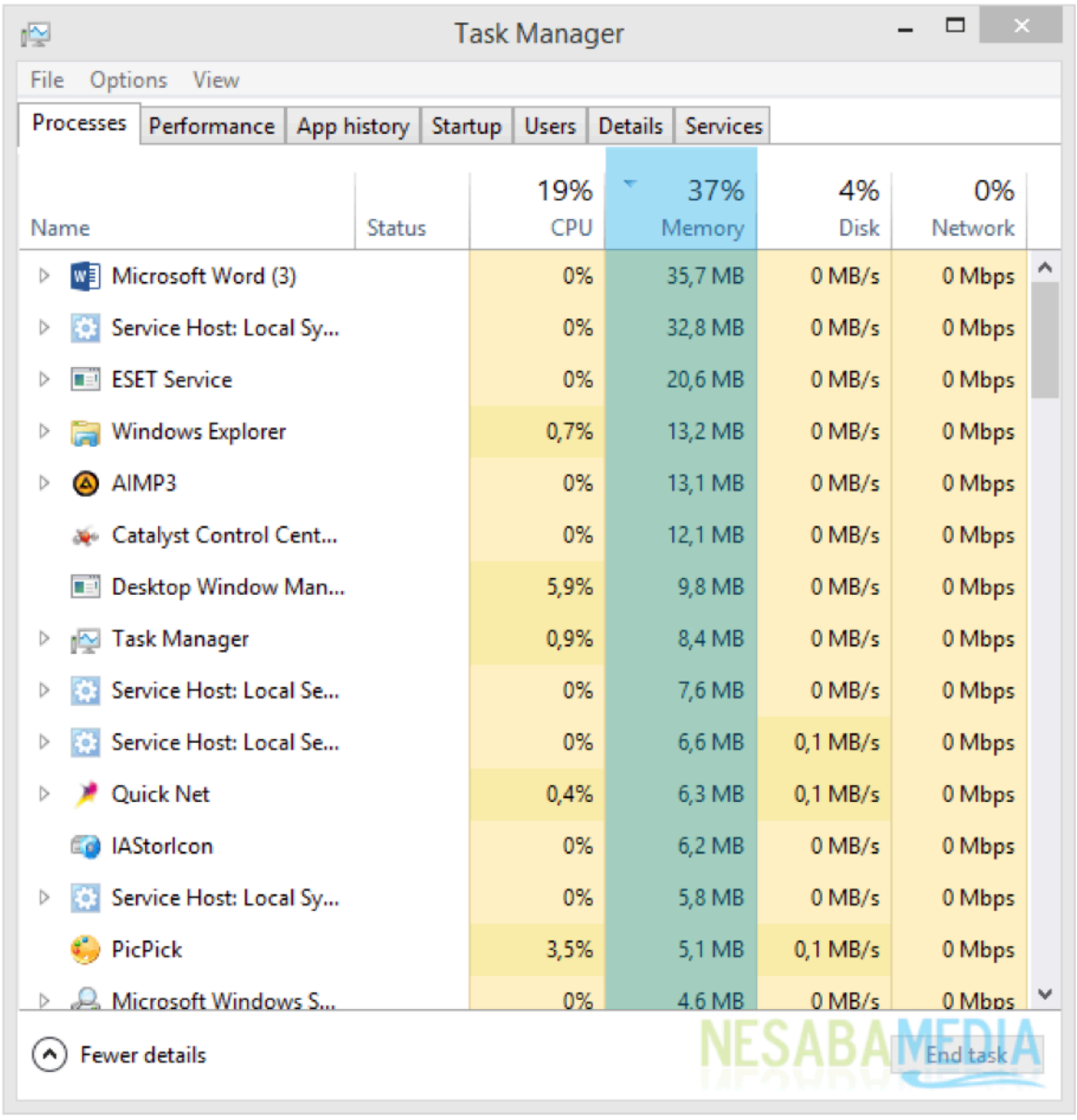
ALAT DAN BAHAN

1. Komputer.
2. Windows OS.
3. Task Manager.

PETUNJUK PRAKTIKUM

1. Buka Microsoft Word.

2. Tuliskan beberapa kata atau kalimat pada Microsoft Word.
3. Buka Task Manager.
4. Pada Tab “Processes” dapat dilihat bahwa pada kolom “Memory” menunjukkan jumlah memori yang digunakan oleh Sistem Komputer untuk menyimpan informasi sementara berupa sejumlah kata pada aplikasi Microsoft Word.



Gambar 2. Tampilan hasil dari Task Manager

5. Dengan menyimpan data secara sementara pada RAM maka program dapat berjalan dengan lebih cepat dan responsive. Mengingat sifat penyimpanan pada RAM adalah sementara, jangan lupa untuk menyimpan data dalam format yang permanen karena kejadian yang tidak diinginkan misalnya listrik mati atau hal-hal lain bisa menyebabkan data hilang dan tidak bisa diakses kembali.

PRAKTIKUM 02 : PEMROGRAMAN ASSEMBLY DENGAN DEBUG

TUJUAN

1. Mahasiswa mampu menjelaskan perintah dalam debug.
2. Mahasiswa mampu membuat program assembly.
3. Mahasiswa mampu menerapkan debug pada program assembly.

DASAR TEORI

Dalam mempelajari bahasa assembly tidak lepas dari register, karena dalam pemograman bahasa assembly selalu terhubung dengan register yang digunakan dalam pemograman untuk melakukan operasi-operasi dan instruksi–instruksi yang diperintahkan pada bahasa assembly.

Register adalah sebagian kecil memory CPU yang dipakai untuk tempat penampungan data. Data yang disimpan dalam register akan diproses dalam berbagai operasi. Besarnya data disesuaikan dengan daya tampung register. Secara umum register dapat dibagi dalam lima golongan yaitu:

1. General Purpose Register.
   Register yg digunakan untuk keperluan umum pemograman, keperluan umum yang dimaksud yaitu melakukan perhitungan aritmetika dan perpindahan data. Register ini terdiri atas:
   a. Register AX
      Register AX merupakan register 16 bit sebagai register aritmatika atau Accumulator Register. Register AX selalu dipakai dalam operasi penjumlahan, pengurangan, perkalian dan pembagian. Register AX terdiri dari dua buah register 8 bit, yaitu AH dan AL.
   b. Register BX
      Register BX merupakan register 16 bit sebagai register base addressing mode, yaitu berfungsi mengambil atau menulis data langsung dari/ke memory disebut Base Address Register. Register BX terdiri dari dua buah register 8 bit yaitu BH dan BL.
   c. Register CX
      Register CX merupakan register 16 bit sebagai suatu counter untuk meletakan jumlah lompatan pada loop-loop yang anda lakukan, disebut Count Register. Register CX terdiri dari dua buah register 8 bit yaitu CH dan CL.
   d. Register DX
      Disebut juga Data Register. Register ini terdiri dari dua buah register 8 bit yaitu DH dan DL. Register DX merupakan register 16 bit yang mempunyai tugas:
      - Membantu AX dalamoperasi perkalian pembagian
      - DX merupakan register offset dari DS
      - Menunjukkan nomor port pada operasi port.
2. Segment Register
   Segment Register membentuk alamat memori bagi suatu data, terdapat pada 2 operasi (Real Mode dan Protected Real), berikut register-register pada segment register:
   a. Register CS : Code Segment Register
   b. Register DS : Data Segment Register
   c. Register SS : Stack Segment Register
   d. Register ES : Extra Segment Register
3. Point Register
   Pointer Register adalah register yg digunakan untuk menunjukkan alamat sebuah data di lokasi memori. Penujukan pada saat operasi perpindahan data, operasi stack (PUSH dan POP), akan dieksekusi pada saat proram dijalankan. Pointer register terdiri atas:
   a. Instruction Point (IP) Register

Instruction Pointer berfungsi sebagai tempat menyimpan alamat dari kode yang akan dieksekusi selanjutnya oleh mikroprosesor. Dalam kerjanya IP bekerja dengan code segmen (CS) untuk menghasilkan alamat relative dari suatu instruksi.

b. Stack Point (SP) Register
c. Base Point (BP) Register

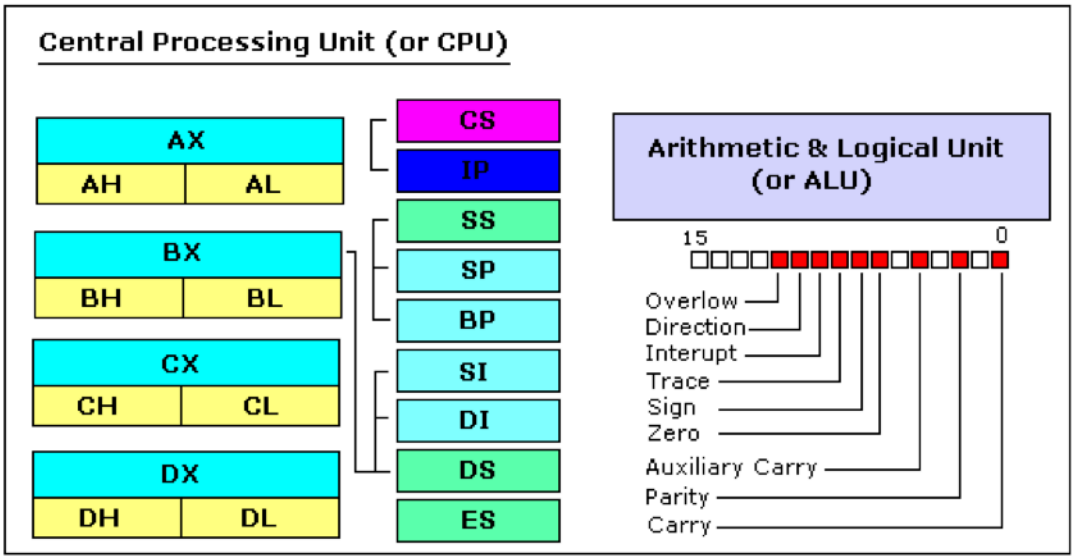
4. Index Register

Index Register digunakan untuk menunjukkan alamat sebuah data di lokasi memori pada operasi-operasi string. Register ini juga digunakan untuk menujukkan data berindek (data tipe larik) di dalam memori, Register ini terdiri atas :

a. Source Index (SI) Register
b. Destination Index (DI) Register

5. Flag Register

Flag Register berfungsi untuk menunjukan status (Keadaan) sesaat dari mikroprocecor. Register ini terdiri atas : overflow, direction, overflow, zero, carry, parity, auxiliary, sign, trap dan Interrupt.



Gambar 3. Macam-macam Register

Perintah-Perintah Debug:

A	:	Merakit instruksi simbolik (kode mesin)
D	:	Menampilkan isi suatu daerah memori
E	:	Memasukan data ke memori yang dimulai pad lokasi tertentu
G	:	Run executable program ke memori
N	:	Memberikan Nama Program
P	:	Eksekusi sekumpulan instruksi yang terkait
Q	:	Quit
R	:	Menampilkan isi satu atau lebih register
T	:	Trace isi sebuah Instruksi
U	:	Unassembled kode mesin ke kode simbolik
W	:	Menulis program ke disk

Perintah Dasar Assembly:

1. MOV.

Perintah MOV adalah perintah untuk mengisi, memindahkan, memperbaruhi isi suatu register, variable ataupun lokasi memory. Adapun tata penulisan perintah MOV adalah :

```
MOV [operand A], [Operand B]
```

Contoh :

```
MOV AH,02
```

Operand A adalah Register AH Operand B adalah bilangan 02
Hal yang dilakukan oleh komputer untuk perintah diatas adalah memasukan 02 ke register AH.

2. INT.
   Perintah INT memanggil subroutine (sub program kecil) yang telah disediakan oleh memory komputer. INT ada dua 2 jenis yaitu:
   a. Interrupt 00h – 1Fh(0 - 31) merupakan interrupt BIOS dan standar di semua komputer baik yang menggunakan sistem operasi DOS maupun bukan yang menggunakan sistem operasi DOS. Lokasi Interrupt Vector Table-nya ada di alamat absolute 0000h-007Fh.
   b. Interrupt 20h–FFh(32-255)merupakaninterruptDOS.Interruptinihanya ada pada komputer yang menggunakan sistem operasi DOS dan interrupt handler-nya dip roses ke memori oleh DOS pada saat DOS digunakan. Lokasi Interrupt Vector Table-nya ada di alamat absolute 07h-3FFh.

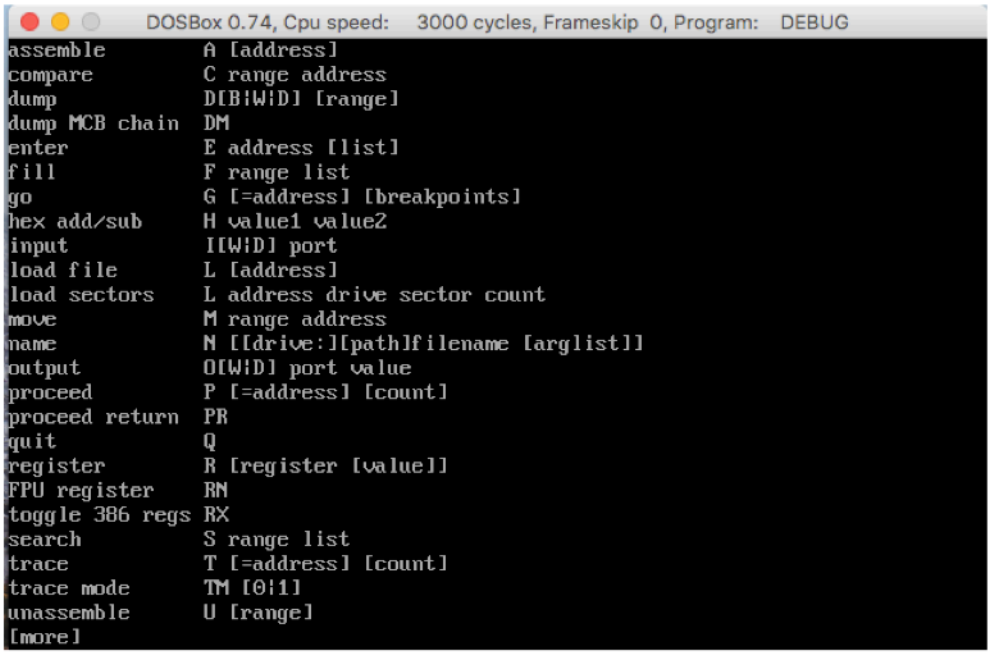
ALAT DAN BAHAN

1. Komputer.
2. Windows OS.
3. Command Prompt.

PETUNJUK PRAKTIKUM

Latihan 1.

1. Buka Command Prompt**.**
2. Ketik “DEBUG” pada halaman Command Prompt kemudian tekan enter.
3. Untuk mengetahui list perintah pada DEBUG, Ketikkan “?” lalu tekan enter.

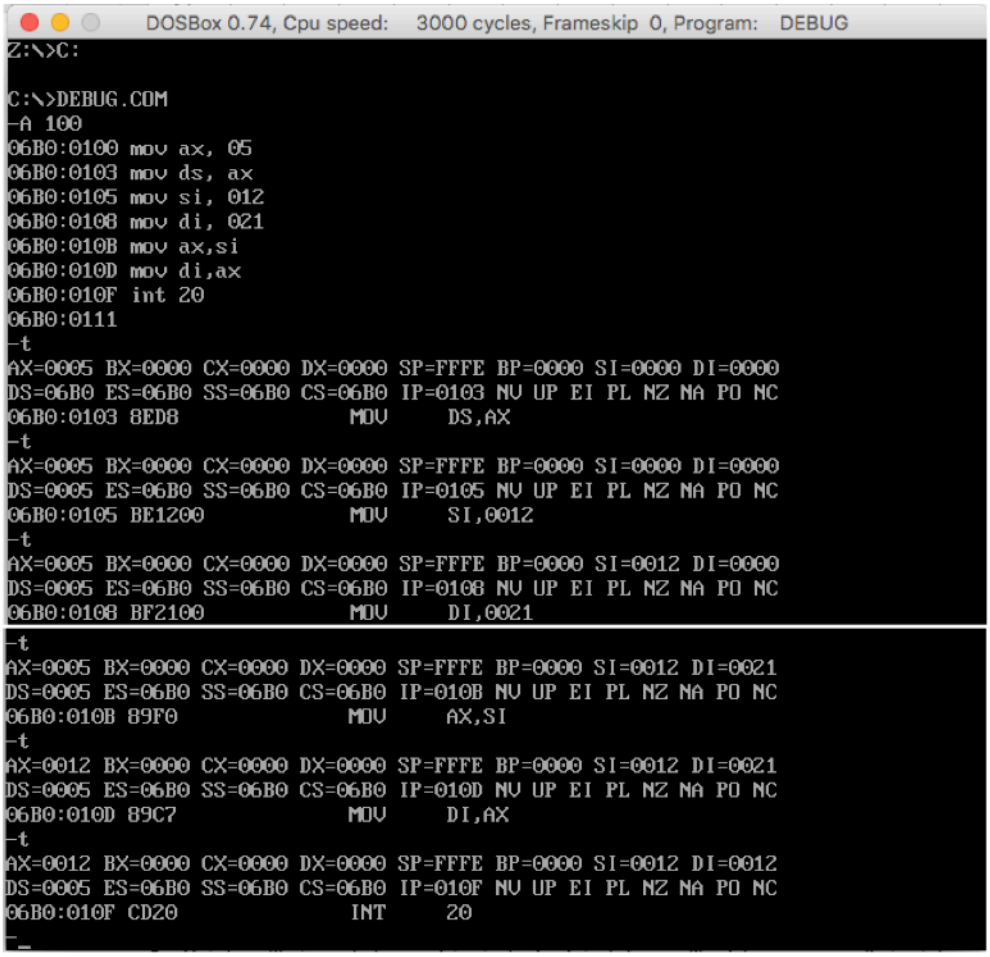


Gambar 4. Tampilan perintah-perintah Debug

4. Ketikkan “A 100” lalu tekan enter.
5. Selanjutnya ketikkan program di bawah ini:

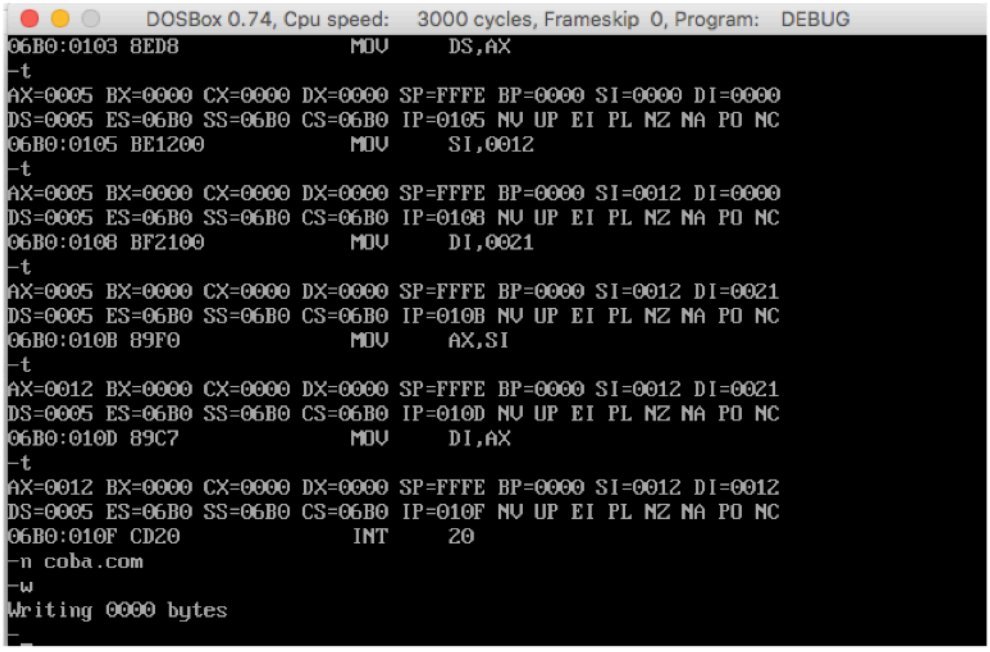
```
Mov ax,05
Mov ds,ax
Mov si,012
Mov di,021
Mov ax,si
Mov di,ax
Int 20
```

6. Untuk melihat perubahan Register terhadap instruksi yang diberikan program di atas tekan enter, ketikan “t” dan kemudian tekan enter.



Gambar 5. Tampilan trace instruksi

7. Untuk menyimpan program ke disk dengan cara mengetikkan huruf “N [nama_program].com” kemudian Ketikan “W” lalu tekan Enter.



Gambar 6. Tampilan penyimpanan program ke dalam disk

Latihan 2.

1. Latihan menulis, menghitung panjang program, memberi nama, menyimpan program, memberi nama dan menjalankan program.
2. Ketikkan “A 100” lalu tekan enter.
3. Selanjutnya ketikkan program di bawah ini:

```
Mov ax,02
Mov dl,41
Int 20
Int 21
```

4. Untuk menentukan Panjang program tekan enter, kemudian ketikkan “RCX” lalu isi angka “8” karena panjangnya program 8 bit, diperoleh dari angka alamat A0108 - A0100 = 8.
5. Untuk menyimpan program, ketikkan huruf “N [nama_program].com” kemudian Ketikan “W” lalu tekan Enter.
6. Untuk menjalankan program ketikkan huruf “G” lalu tekan enter, Maka akan terlihat hasil berupa huruf “A”.
7. Ketikkan “Q” untuk keluar dari DEBUG lalu tekan enter.
8. Untuk mengetahui Panjang program, ketikkan “dir nama_program.com” lalu tekan enter.

Latihan 3.

1. Ketikkan “A 100” lalu tekan enter.
2. Selanjutnya ketikkan program di bawah ini:

```
Mov ah,02
Mov dl,31
Int 20
Int 21

N nama_program.com
W
G
```

3. Hasilnya berupa angka “1”

EVALUASI (POST TEST)

1. Analisis Program pada Latihan 1, 2, dan 3 dengan menggunakan trace. Jelaskan perubahan yang terjadi!
2. Ubahlah **Latihan 1** dengan menukar data register DI ke Register SI, kemudian hapus data pada Register AX.

PRAKTIKUM 03 : PEMROGRAMAN COMPILER ASSEMBLY DENGAN TASM DAN TLINK

TUJUAN

1. Mahasiswa mampu menjelaskan tentang konsep tanda *directive*.
2. Mahasiswa mampu menjelaskan tentang konsep assembly compiler Tasm dan Tlink.
3. Mahasiswa mampu menguasai assembly compiler Tasm dan Tlink.
4. Mahasiswa mampu menguasai penulisan program assembly dengan text editor.
5. Mahasiswa mampu meng-compile program asm menjadi com dan exe.

DASAR TEORI

Program assembly adalan kumpulan dari baris-baris text program. Baris-baris program ini dapat ditulis dengan text editor secara umum seperti text editor dari window maupun text editor khusus mengedit program seperti note++.

Struktur Program Assembly

Struktur program adalah aturan penulisan program agar nantinya dapat diterima dan dapat di-compile. Struktur program assembly ini akan compile dengan TASM dan TLINK.

```
------------------------------------------------------------
.MODEL SMALL
.CODE
ORG 100H
Label1 : JMP Label2
+---------------------+
| TEMPAT DATA PROGRAM |
+---------------------+
Label2 : +---------------------+
| TEMPAT PROGRAM |
+---------------------+
INT 20H
END Label1
------------------------------------------------------------
```

Tanda Directive

1. .MODEL. Model-model pada program dan codenya:
    a. TINY, program hanya akan menggunakan 1 segment seperti program COM, model ini disediakan untuk program COM.
    b. SMALL, data dan code yang digunakan oleh program kurang dari ukuran 1 segment atau 64 KB.
    c. MEDIUM, data yang digunakan oleh program kurang dari 64 KB tetapi code yang digunakan bisa lebih dari 64 KB.

d. COMPACT, data yang digunakan bisa lebih besar dari 64 KB tetapi codenya kurang dari 64 KB.
e. LARGE, data dan code yang dipakai oleh program bisa lebih dari 64 KB.
f. HUGE, data maupun code array yang digunakan bisa lebih dari 64 KB.

2. .CODE
   Tanda directive yang digunakan untuk memberikan pemberitahuan mengenai code segmentnya. Code segment ini digunakan untuk menyimpan program yang akan dijalankan.
3. .ORG 100h
   Tanda directive ini sering digunakan pada program COM. Perintah ini digunakan untuk memberitahukan kepada assembler supaya program pada saat dijalankan ditaruh mulai pada offset ke 100h (256) byte. Dapat diartikan juga bahwa kita menyediakan 100h byte kosong pada saat program dijalankan. Program kosong ini nantinya akan ditempati oleh PSP (*Program Segment Prefix)*.
4. .JMP
   Perintah JMP digunakan untuk melompat menuju tempat yang ditunjukkan oleh perintah JUMP.

   Syntax:
   ```
   JUMP Tujuan
   ```

5. INT 20h
   Perintah ini digunakan untuk megakhiri program dan menyerahkannya kembali ke DOS.

Meng-compile program assembly dengan TASM dan TLINK

Ada dua tahap untuk meng-compile program assembly, pertama adalah meng-assembly dengan TASM. Menga-assembly artinya mengubah text program menjadi data objek berupa data biner. Caranya sebagai berikut:

1. Menggunakan perintah TASM diikuti nama_program.asm

   ```
   TASM <spasi> file.asm
   ```

   Contoh
   ```
   TASM prog0201.asm <enter>
   ```

   Jika proses assembly ini tidak ada pesan error, maka proses assembly berhasil, dan menghasilkan file OBJ, yaitu prog01.obj
2. Melakukan link agar data biner tersebut dapat dieksekusi menjadi bertipe COM atau EXE dengan cara:

   ```
   TLINK prog01.obj -t
   ```
   Atau
   ```
   TLINK prog01.obj/t
   ```

   Maka akan diperoleh file prog01.map dan prog01.com. File yang dapat dieksekusi adalah yang bertipe com, yaitu prog01.com

3. Mengeksekusi program yaitu dengan memanggil program bertipe com, yitu prog01.com. Caranya

   ```
   Prog01.com <enter>
   ```
   Atau
   ```
   Prog01 <enter>
   ```

Demikian penjelasan tentang struktur program assembly dan tahap kompilasi dengan TASM dan TLINK. File OBJ dan MAP adalah file garbage (sampah) yang selanjutnya harus dihapus.

Batch File

Untuk menyingkat proses kompilasi dapat dibuat batch file yang berisi beberapa perintah yang akan dilaksanakan secara otomatis.

1. Buka editor, tulis baris-baris perintah berikut:

```
@echo off cls
echo Sedang meng-compile file ASM menjadi COM
echo ------------------------------------------------------------
---
echo oleh Taufiq Ismail

tasm %1
tlink %1 -t

del %1.map del %1.obj

echo Selesai...
echo Jika tidak ada pesan error, coba eksekusi program tersebut
```

2. Simpan dengan nama file `ASM2COM.BAT`
3. Jalankan dengan menggunakan perintah berikut:

```
Asm2com <spasi> prog01 <enter>
```

Perhatikan, tipe file .ASM tidak disertakan dalam perintah tersebut. Selamat berlatih dan mencoba

Intrupsi (Mencetak Karakter Dan Kalimat)

Intrupsi adalah permitaan khusus kepada mikroprosesor untuk melakukan sesuatu. Mikroprosesor akan menghentikan dahulu apa sedang dikerjakan dan melayani permitaan khusus tersebut, bila terjadi intrupsi. Ada 2 jenis intrupsi yaitu intrupsi vector (yang dimilki mikroprosesor) dan instruksi intrupsi (dimiliki PC).

Ada banyak intrupsi yg dimiliki oleh mikroprosesor dan PC tapi pada praktikum ini hanya terfokus pada intrupsi 20 dan 21.

1. Intrupsi 20 hanya memilki 1 fungsi layanaan yaitu untuk menghentikan program
2. Intrupsi 21 terdiri atas 2 layanan (*service*) yaitu:
    a. *Service* nomor 02h berfungsi mencetak karakter ASCII
    b. *Service* nomor 09h berfungsi mencetak String ASCII.

Untuk memanggil layanan nomor intrupsi harus dimasukan dulu ke register AH sebelum INT.

ALAT DAN BAHAN

1. Komputer.
2. Text Editor (Notepad/Notepad++).
3. Command Prompt.

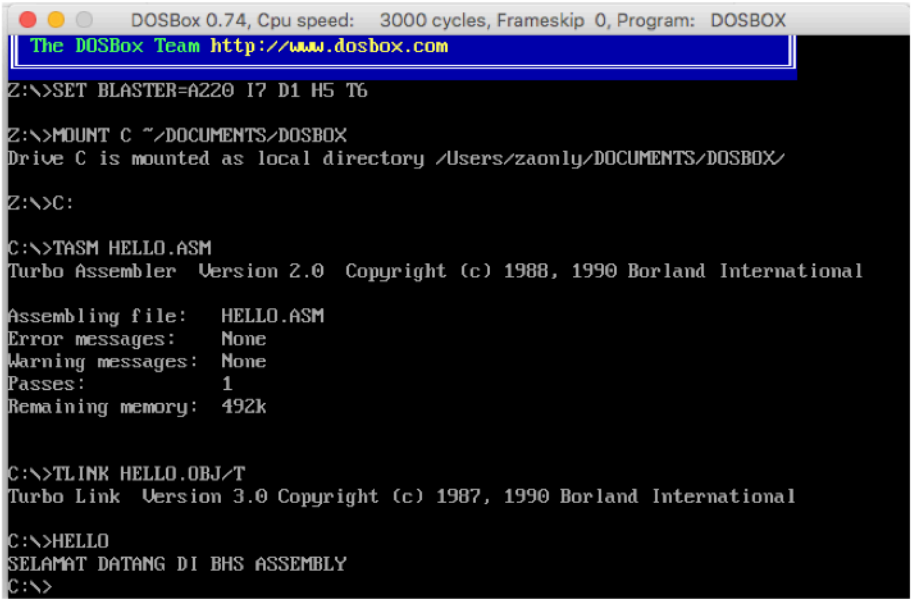
PETUNJUK PRAKTIKUM

Latihan 1.

1. Buka Text Editor (Notepad).
2. Ketikkan program di bawah ini:

```
.MODEL SMALL
.CODE
ORG 100h
MULAI:
   jmp CETAK
   Hello DB 'SELAMAT DATANG '
         DB 'DI BHS ASSEMBLY'
         DB '$'
CETAK :
   MOV AH,09H
   MOV DX,OFFSET Hello
   INT 21H
HABIS:
   INT 20h
END MULAI
```

3. Simpan Program yang anda ketik di atas dengan nama program `Hello.asm`.
4. Gunakan perintah tasm untuk mengubah ke file objek.
   `C:/tasm>tasm Hello.asm`
5. Gunakan perintah tlink untuk merubah ke file Com.
   `C:/tasm>tlink Hello.obj/t`
6. Ketikan nama program untuk menjalankan program comnya
   `C:/tasm>Hello`
7. Hasil program berupa: `SELAMAT DATANG DI BAHASA ASSEMBLY`



Gambar 7. Tampilan hasil program assembly latihan 1.

Latihan 2.

1. Ketikkan program di bawah ini:

```
.MODEL SMALL
.CODE
ORG 100h
```

```
Proses :
   MOV AH,02h
   MOV DL,'A'
   INT 21h
   INT 20h
END Proses
```

2. Simpan Program yang anda ketik di atas dengan nama program `Latihan1.asm`.
3. Gunakan perintah TASM dan TLINK.
4. Jalankan program.

Latihan 3.

1. Ketikkan program di bawah ini:

```
.MODEL SMALL
.CODE
ORG 100h
Satu:
   JMP Dua
   Kal1  DB 'Saya Lagi Belajar'
         DB 'Bahasa Assembly $'
   Kal2  DB 'Ternyata...'
         DB 'Asik...'
         DB '$'
Dua :
   Mov AH,09h
   Lea DX,Kal1
   int 21h
Tiga :
   Mov DX,Offset Kal2 int 21h
   int 20h
End Satu
```

2. Simpan Program yang anda ketik di atas dengan nama program `Latihan2.asm`.
3. Gunakan perintah TASM dan TLINK.
4. Jalankan program.

EVALUASI (POST TEST)

1. Dengan menggunakan Latihan 1 sebagai Referensi, Buatlah program untuk menampilkan karakter ‘ K’.
2. Dengan menggunakan Latihan 2 sebagai Referensi, Buatlah program untuk menampilkan karakter ‘ PRODI INFORMATIKA’.

PRAKTIKUM 04 : PEMROGRAMAN ASSEMBLY DENGAN WORD PROCESS

TUJUAN

1. Mahasiswa mampu menjelaskan tentang konsep word process.
2. Mahasiswa mampu menerapkan pemrograman assembly dengan word process.

DASAR TEORI

Dalam pemerograman assembler, interupsi merupakan hal yang penting untuk menghasilkan sesuatu yang diinginan oleh user sehingga dikenali oleh komputer. Interupsi ini tidak berjalan sendiri, interupsi selalu berjalan berbarengan dengan service yang menyediakan fungsi-fungsi yang berbeda untuk setiap interupsi yang sama tetapi dengan service yang berbeda-beda.
Dalam mencetak kalimat dengan fungsi DOS, interupsi yang digunakan adalah interrupt 21h dengan service 09h. Pemakaian interrupt 21h dengan service 09h ini dapat digunakan untuk mencetak kalimat dengan aturan:

```
INPUT
AH = 9
DS:DX = Alamat String ($) yang diinginkan
```

Catatan bahwa karakter $ dijadikan tanda akhir dalam tulisan. Khusus untuk mencetak kalimat ini, $ merupakan akhir pada kalimat yang sering diganakan untuk mengakhiri pendefinisian dari sebuah string, sehingga kalimat tersebut akan dicetak sampai tanda $. Karakter $ dijadikan sebagai tanda akhir pada string sehingga string yang akan dicetak adalah string yang berada disebelum tanda $ dan selainnya yang berada disesudah tanda $ tidak akan dicetak. Selain tanda $ yang digunakan untuk mengakhiri kalimat, ada juga tanda 10 dan tanda 13 yang masing-masingnya digunakan untuk pindah baris dan menuju kolom 0 kembali.
Sebuah karakter disertai dengan warna tentunya akan lebih menarik. Untuk itu anda bisa menggunakan interupsi ke 10h dengan aturan pemakaiannya:

```
INPUT
AH = 09h
AL = Kode ASCII dari karakter yang akan dicetak
BH = Nomor halaman (0 untuk halaman 1)
BL = Atribut atau warna dari karakter yang akan dicetak
CX = Banyaknya karakter tersebut akan dicetak
```

Karakter Kontrol merupakan karakter yang digunakan sebagai kontrol dari program khusus mencetak kalimat. Karakter kontrol ini digunakan untuk keperluan komunikasi komputer dengan periferalnya. Karakter kontrol yang sering digunakan dapat dilihat pada Tabel 1.

Tabel 1. Karakter control yang sering digunakan

Code	Nama	Fungsi
07	Bel	Memberikan suara BEEP
08	Backspace (BS)	Memindahkan kursor 1 kolom ke belakang
09	Horizontal Tab	Memindahkan kursor 8 kolom ke kanan
10	Line Feed (LF)	Memindahkan kurosr 1 baris ke bawah
13	Carriage Return (CR)	Memindahkan kursor menuju awal baris

Selain karakter kontrol pada Tabel 1, karakter kontrol yang tersedia pada ASCII secara lengkap dapat dilihat pada Tabel 2.

Tabel 2. Karakter Kontrol pada ASCII

Code	Nama
0	Nul
1	Start of heading
2	Start of text
3	End of text
4	End of transmission
5	Enquiry
6	Acknowledge
7	Bel
8	Backspace
9	Horizontal tabulation
10	Line feed
11	Vertical tabulation
12	Form feed
13	Carriage return
14	Shift out
15	Shift in
16	Data link escape
17	Device control
18	Negative acknowledge
19	Synchronous table
20	End of transmission block
21	Cancel
22	End of medium
23	Substitute
24	Escape
25	File separator
26	Group separator
27	Record separator
28	Unit separator
29	Space
30	Delete

ALAT DAN BAHAN

1. Komputer.
2. Text Editor.
3. Command Prompt.

PETUNJUK PRAKTIKUM

Latihan 1.

1. Buka Text Editor.
2. Ketikkan code program di bawah ini:

```
.MODEL SMALL
.CODE
ORG 100h
Proses:
   MOV AH,09h       ; Nilai servis untuk mencetak karakter
```

```
   MOV AL,'A'       ; AL = Karakter yang akan dicetak
   MOV BH,00h       ; Nomor Halaman layar
   MOV BL,93h       ; Warna atau atribut dari karakter
   MOV CX,03h       ; Banyaknya karakter yang ingin dicetak
   INT  10h         ; Laksanakan!!!

   INT 20h          ; Selesai! kembali ke DOS
END Proses
```

3. Simpan Program yang anda ketik di atas dengan `nama_program.asm`.
4. Gunakan perintah TASM dan TLINK.
5. Jalankan program.
6. Hasilnya adalah huruf "A" sebanyak 3 kali dengan warna dasar biru kedip dan warna tulisan Cyan.

Latihan 2.

1. Buka Text Editor.
2. Ketikkan code program di bawah ini:

```
.MODEL SMALL
.CODE
ORG 100h
Proses:
   MOV AH,02h       ; Nilai servis
   MOV DL,'A'       ; DL=karakter 'A' atau DL=41h
   MOV CX,10h       ; Banyaknya pengulangan yang akan
Ulang:
   INT 21h          ; Cetak karakter!!
   INC DL           ; Tambah DL dengan 1
   LOOP Ulang       ; Lompat ke Ulang

   INT  20h
END      Proses
```

3. Simpan Program yang anda ketik di atas dengan `nama_program.asm`.
4. Gunakan perintah TASM dan TLINK.
5. Jalankan program.
6. Hasilnya adalah "ABCDEFGHIJKLMNOP"

Latihan 3.

1. Buka Text Editor.
2. Ketikkan code program di bawah ini:

```
.model SMALL
.code
       ORG 100h

tdata:  jmp proses
        lusername     db 13,10,'Username : $'
        lpassword     db 13,10,'Password : $'
        lditerima     db 13,10,'diterima $'
        lditolak      db 13,10,'ditolak $'

        ; menyiapkan memori (var) untuk menerima inputan keyboard
        vusername     db 23,?,23 dup(?)
        vpassword     db 23,?,23 dup(?)

proses:
```

```
        mov ah,09h          ; tampilkan tulisan username
        lea dx,lusername
        int 21h

        mov ah,0ah          ; menunggu masukkan keyboard
        lea dx,vusername
        int 21h

        mov ah,09h            ; tampilkan tulisan password
        lea dx,lpassword
        int 21h

        mov ah,0ah
        lea dx,vpassword
        int 21h

        lea si,vusername
        lea di,vpassword

        cld
        mov cx,23
        rep cmpsb
        jne gagal

        mov ah,09h
        lea dx,lditerima
        int 21h
        jmp exit

gagal :
        mov ah,09h
        lea dx,lditolak
        int 21h
        jmp proses

exit :
        int 20h

end tdata
```

5. Simpan Program yang anda ketik di atas dengan `nama_program.asm`.
6. Gunakan perintah TASM dan TLINK.
7. Jalankan program

EVALUASI (POST TEST)

1. Analisis Program pada Latihan 1, 2, dan 3 dengan menggunakan trace. Jelaskan perubahan yang terjadi!
2. Ubahlah **Latihan 3** dengan menukar data `lditolak` dan `lditerima`.

PRAKTIKUM 05 : ARITMATIKA

TUJUAN

1. Mahasiswa mampu menjelaskan konsep Aritmatika pada pemrograman Assembly.
2. Mahasiswa mampu menerapkan penambahan, pengurangan, perkalian, dan pembagian dengan menggunakan Bahasa pemrograman Assembly.

DASAR TEORI

Operasi Penambahan

1. ADD

   Untuk menambah dalam bahasa assembler digunakan perintah ADD dan ADC serta INC. Perintah ADD digunakan dengan syntax :

   ```
   ADD Tujuan,Asal
   ```

   Perintah ADD ini akan menambahkan nilai pada Tujuan dan Asal. Hasil yang didapat akan ditaruh pada Tujuan, dalam bahasa pascal sama dengan instruksi

   ```
   Tujuan:=Tujuan + Asal.
   ```

   Sebagai contohnya :

   ```
   MOV AH,15h    ; AH:=15h
   MOV AL,4      ; AL:=4
   ADD AH,AL     ; AH:=AH+AL, jadi AH=19h
   ```

   Perlu diperhatikan bahwa pada perintah ADD ini antara Tujuan dan Asal harus mempunyai daya tampung yang sama, misalnya register AH(8 bit) dan AL(8 bit), AX(16 bit) dan BX(16 bit).

2. ADC

   Perintah ADC digunakan dengan cara yang sama pada perintah ADD, yaitu :

   ```
   ADC Tujuan,Asal
   ```

   Perbedaannya pada perintah ADC ini Tujuan tempat menampung hasil pertambahan Tujuan dan Asal ditambah lagi dengan carry flag (Tujuan:=Tujuan+Asal+Carry). Pertambahan yang demikian bisa memecahkan masalah seperti yang pernah kita kemukakan, seperti pertambahan pada bilangan 12345678h+9ABCDEF0h. Seperti yang telah kita ketahui bahwa satu register hanya mampu menampung 16 bit, maka untuk pertambahan seperti yang diatas bisa anda gunakan perintah ADC untuk memecahkannya, Contoh:

   ```
   MOV AX,1234h  ; AX = 1234h CF = 0
   MOV BX,9ABCh  ; BX = 9ABCh CF = 0
   MOV CX,5678h  ; BX = 5678h CF = 0
   MOV DX,0DEF0h ; DX = DEF0h CF = 0
   ADD CX,DX           ; CX = 3568h CF = 1
   ADC AX,BX           ; AX = AX+BX+CF = ACF1
   ```

Hasil penjumlahan akan ditampung pada register AX:CX yaitu ACF13568h. Adapun flag-flag yang terpengaruh oleh perintah ADD dan ADC ini adalah CF, PF, AF, ZF, SF dan OF.

3. INC

Perintah INC(Increment) digunakan khusus untuk pertambahan dengan 1. Perintah INC hanya menggunakan 1-byte memory, sedangkan perintah ADD dan ADC menggunakan 3 byte. Oleh sebab itu bila anda ingin melakukan operasi pertambahan dengan 1 gunakanlah perintah INC. Syntax pemakainya adalah:

```
INC Tujuan
```

Nilai pada tujuan akan ditambah dengan 1, seperti perintah Tujuan:=Tujuan+1 dalam Turbo Pascal. Tujuan disini dapat berupa suatu register maupun memory. Contoh:

```
INC AL   ; AL=AL+1.
```

Adapun flag yang terpengaruh oleh perintah ini adalah OF,SF,ZF,AF dan PF.

Operasi Pengurangan

1. SUB

Untuk Operasi pengurangan dapat digunakan perintah SUB dengan syntax:

```
SUB Tujuan,Asal
```

Hasil yang didapat akan ditaruh pada Tujuan, dalam bahasa pascal sama dengan instruksi

```
Tujuan:=Tujuan-Asal.
```

Contoh :

```
MOV AX,15 ; AX:=15
MOV BX,12 ; BX:=12
SUB AX,BX ; AX:=15-12=3
SUB AX,AX ; AX=0
```

Untuk menolkan suatu register bisa anda kurangkan dengan dirinya sendiri seperti SUB AX,AX

2. SBB

Seperti pada operasi penambahan, maka pada operasi pengurangan dengan bilangan yang besar(lebih dari 16 bit), bisa anda gunakan perintah SUB disertai dengan SBB(Substract With Carry). Perintah SBB digunakan dengan syntax:

```
SBB Tujuan,Asal
```

Perintah SBB akan mengurangkan nilai Tujuan dengan Asal dengan cara yang sama seperti perintah SUB, kemudian hasil yang didapat dikurangi lagi dengan Carry Flag (Tujuan:=Tujuan- Asal-CF).

3. DEC

Perintah DEC (*Decrement*) digunakan khusus untuk pengurangan dengan 1. Perintah DEC hanya menggunakan 1-byte memory, sedangkan perintah SUB dan SBB menggunakan 3 byte. Oleh sebab itu bila anda ingin melakukan operasi pengurangan dengan 1 gunakanlah perintah DEC. Syntax pemakaian perintah dec ini adalah:

```
DEC Tujuan
```

Nilai pada tujuan akan dikurangi 1, seperti perintah Tujuan:=Tujuan-1 dalam Turbo Pascal. Tujuan disini dapat berupa suatu register maupun memory. Contoh:

```
DEC AL   ; AL=AL-1
```

Operasi Perkalian

4. MUL

Untuk perkalian bisa digunakan perintah MUL dengan syntax:

```
MUL Sumber
```

Sumber disini dapat berupa suatu register 8-bit (Misal: BL,BH,..), register 16-bit (Misal: BX,DX,..) atau suatu varibel. Ada 2 kemungkinan yang akan terjadi pada perintah MUL ini sesuai dengan jenis perkalian 8-bit atau 16-bit.

Bila Sumber merupakan 8-bit seperti MUL BH maka komputer akan mengambil nilai yang terdapat pada BH dan nilai pada AL untuk dikalikan. Hasil yang didapat akan selalu disimpan pada register AX. Bila sumber merupakan 16-bit seperti MUL BX maka komputer akan mengambil nilai yang terdapat pada BX dan nilai pada AX untuk dikalikan. Hasil yang didapat akan disimpan pada register DX dan AX(DX:AX), jadi register DX menyimpan Word tingginya dan AX menyimpan Word rendahnya.

Contoh:

```
.MODEL SMALL
.CODE
ORG 100h
Tdata:
   JMP Proses
   A     DW    01EFh
   B     DW    02FEh
   HslLo DW    ?
   HslHi DW    ?
Proses:
   MOV   AX,A        ; AX=1EF
   MUL   B           ; Kalikan 1EF*2FE
   MOV   HslLo,AX    ; AX bernilai C922 sehingga HslLo=C922
   MOV   HslHi,DX    ; DX bernilai 0005 sehingga HslHi=0005

   INT   20h         ; Kembali ke DOS
END       Tdata
```

Operasi Pembagian

5. DIV

Operasi pada pembagian pada dasarnya sama dengan perkalian. Untuk operasi pembagian digunakan perintah DIV dengan syntax:

```
DIV Sumber
```

Bila sumber merupakan operand 8-bit seperti DIV BH, maka komputer akan mengambil nilai pada register AX dan membaginya dengan nilai BH. Hasil pembagian 8-bit ini akan disimpan pada register AL dan sisa dari pembagian akan disimpan pada register AH.

Bila sumber merupakan operand 16-bit seperti DIV BX, maka komputer akan mengambil nilai yang terdapat pada register DX:AX dan membaginya dengan nilai BX. Hasil pembagian 16-bit ini akan disimpan pada register AX dan sisa dari pembagian akan disimpan pada register DX.

Contoh:

```
.MODEL SMALL
.CODE
ORG 100h
Tdata:
   JMP Proses
   A     DW    01EFh
   B     DW    02FEh
   HslLo DW    ?
   HslHi DW    ?
Proses:
   SUB   DX,DX       ;
   MOV   AX,A        ; AX=1EF
   DIV   B           ; Bagi 1EF:2
   MOV   Hsl,AX      ; AX bernilai 00F7 sehingga Hsl=00F7
   MOV   Sisa,DX     ; DX berisi 0001 sehingga Sisa=0001

   INT   20h         ; Kembali ke DOS
END      Tdata
```

ALAT DAN BAHAN

1. Komputer.
2. Text Editor.
3. Command Prompt.

PETUNJUK PRAKTIKUM

1. Buka Command Prompt.
2. Ketikkan “Debug” lalu tekan Enter.
3. Ketikkan huruf “A100” lalu tekan enter.
4. Selanjutnya ketikkan program di bawah ini:

```
MOV AH,15h
MOV AL,4
ADD AH,AL
MOV AX,1234h
MOV BX,0F221h
ADD AX,BX
MOV AX,1234h
MOV BX,9ABCh
MOV CX,5678h
MOV DX,0DEF0h
ADD CX,DX
ADC AX,BX
INC AL
INT 20h
```

5. Untuk melihat perubahan Register terhadap instruksi yang diberikan program di atas tekan enter, ketikan “t” dan kemudian tekan enter.
6. Untuk menyimpan program ke disk dengan cara mengetikkan huruf “N [nama_program].com” kemudian Ketikan “W” lalu tekan Enter.

EVALUASI (POST TEST)

1. Analisis Program di atas dengan menggunakan trace. Jelaskan perubahan yang terjadi!
2. Ubahlah program di atas untuk register `AH` 20 dan register `AL` 10, dan tentukan hasil register `AL` terakhir.

PRAKTIKUM 06 : MOUSE

TUJUAN

1. Mahasiswa mampu menjelaskan tentang konsep mouse.
2. Mahasiswa mampu menerapkan pemrograman assembly dengan mouse.

DASAR TEORI

Mouse dikendalikan menggunakan fungsi mouse di interupsi 33 jam. Ada banyak fungsi, tetapi bagian ini hanya akan mencakup set dasar yang diperlukan untuk menyelesaikan segala sesuatunya. Untuk informasi lebih lanjut, lihat referensi di halaman web.

Untuk menggunakan mouse, terlebih dahulu memanggil Fungsi 0000h (Reset Driver dan Baca Status). Ini menginisialisasi driver dan perangkat keras. Kursor mouse awalnya akan disembunyikan, jadi harus menggunakan Fungsi 0001h (Tampilkan Mouse Cursor) untuk membuatnya terlihat. Sejak saat itu, cukup hubungi Fungsi 0003h (Posisi Kembali dan Status Tombol) untuk mendapatkan posisi dan status tombol setiap kali program memerlukannya. Pastikan menyembunyikan kursor mouse sebelum program Anda keluar.

Fungsi ini menginisialisasi perangkat keras dan perangkat lunak sehingga mouse siap digunakan. Mouse awalnya akan disembunyikan.

```
Inputs
      AX = 0000h
Outputs
      AX = Status
      0000h : Error. Hardware/software not installed.
      FFFFh : OK. Hardware/software installed.
      BX = Number of buttons
      FFFFh : Two buttons.
      0000h : Other than two buttons.
      0003h : Three
```

Menampilkan Cursor Mouse

Fungsi ini membuat kursor mouse terlihat di layar. Jika Anda memprogram teks atau grafik dengan menulis langsung ke memori video, Anda harus menyembunyikan kursor mouse sebelum melakukannya untuk menghentikan mouse meninggalkan sampah grafis di layar.

```
Inputs
      AX = 0001h
```

Menampilkan Cursor Mouse

Fungsi ini membuat kursor mouse tidak terlihat. Beberapa panggilan ke fungsi ini memerlukan beberapa panggilan ke Fungsi 0001h (Tampilkan Mouse Cursor) sebelum kursor mouse akan muncul lagi, karena driver mouse terus menghitung berapa kali mouse telah disembunyikan.

```
Inputs
      AX = 0002h
```

Status Button

Fungsi ini mengembalikan posisi kursor mouse saat ini dan status tombol. Posisi diukur dalam piksel, dengan titik asal (0,0) di sudut kiri atas layar. Dalam mode teks, setiap karakter diasumsikan oleh driver mouse untuk sesuai dengan delapan piksel secara horizontal dan delapan piksel secara vertikal. Jadi, untuk mendapatkan posisi baris dan kolom kursor mouse dalam mode teks, bagilah nilai dalam CX dan DX dengan delapan.

```
Inputs
      AX = 0003h

Outputs

      BX = Button status (1 = corresponding button pressed)

            Bit 0 : Left mouse button.

            Bit 1 : Right mouse button.

            Bit 2 : Middle mouse button (if present).

            Bits 3-15 : Cleared to 0.

      CX = Pixel column position.

      DX = Pixel row position.
```

Position Mouse Cursor

Fungsi ini akan memposisikan kursor mouse di layar. Seperti dalam fungsi 0003h, posisi diukur dalam piksel, dengan titik asal (0,0) di sudut kiri atas layar. Lihat deskripsi fungsi 0003h untuk informasi lebih lanjut.

```
Inputs
      AX = 0004h

      CX = Column position

      DX = Row position
```

Define Horizontal Cursor Range

Fungsi ini akan membatasi posisi horizontal kursor mouse ke bagian yang ditentukan di layar. Posisi kolom diberikan dalam piksel.

```
Inputs
      AX = 0007h

      CX = Batas kolom paling kiri

      DX = Batas kolom paling kanan
```

Define Vertical Cursor Range

Fungsi ini akan membatasi posisi vertikal kursor mouse ke bagian yang ditentukan di layar. Posisi baris diberikan dalam piksel.

```
Inputs
      AX = 0008h
      CX = Batas baris atas
      DX = Batas baris bawah
```

Mengubah Kursor Mouse

Fungsi ini akan mendefinisikan ulang tampilan kursor mouse saat layar berada dalam mode grafis.

```
Inputs
      AX = 0009h
      CX = Kolom hotspot kursor dalam bitmap (-16 hingga 16)
      DX = Baris hotspot kursor dalam bitmap (-16 hingga 16)
      ES:DX = Pointer ke kursor bitmap
```

Hotspot adalah istilah yang diberikan ke lokasi piksel dalam gambar kursor mouse yang koordinatnya pada layar sama dengan posisi kursor mouse. Pada dasarnya, hot spot ini memungkinkan kita untuk mengetahui dimana seluruh gambar berada di layar relatif terhadap posisi mouse (dikembalikan dalam fungsi 0003h, misalnya). Awalnya, hot spot ada di sudut kiri atas kursor mouse default (panah).

Bitmap kursor dapat berupa gambar 16x16 piksel yang didefinisikan dalam memori sebagai berikut:

Tabel 3. Tabel bitmap kursor

Offset	Size	Description
00h	16 words	Screen Mask
20h	16 words	Cursor Mask

Setiap kata mendefinisikan enam belas piksel berturut-turut, dengan pixel paling kanan menjadi bit paling signifikan. Gambar didefinisikan dimulai dengan deretan teratas piksel dalam gambar.

Gambar terbentuk pada layar dengan pertama ANDing piksel pada layar dengan gambar Screen Mask, lalu XOR piksel pada layar dengan gambar Mask Kursor.

ALAT DAN BAHAN

1. Komputer.
2. Text Editor
3. Command Prompt.

PETUNJUK PRAKTIKUM

1. Buka Command Prompt.
2. Ketikkan Code Program di bawah ini:

```
.MODEL SMALL
.CODE
        org 100h

proses:
        jmp start
        oldX dw -1
```

```
        oldY dw 0

start:
       ; ubah ke mode grafik
        mov ah, 00
        mov al, 13h       ; layar 256 warna, berkukuran 320x200
pixel.
        int 10h           ;

        ; reset mouse status saat ini:
        mov ax, 0
        int 33h
        cmp ax, 0

        ; tampilkan kursor mouse
        mov ax, 1
        int 33h

check_mouse_button:
        mov ax, 3
        int 33h
        shr cx, 1       ; x/2 - in this mode the value of CX is
doubled.
        cmp bx, 1
        jne xor_cursor
        mov al, 1010b   ; pixel color
        jmp draw_pixel

xor_cursor:
        cmp oldX, -1
        je not_required
        push cx
        push dx
        mov cx, oldX
        mov dx, oldY
        mov ah, 0dh     ; get pixel.
        int 10h

        xor al, 1111b   ; pixel color
        mov ah, 0ch     ; set pixel
        int 10h
        pop dx
        pop cx

not_required:
        mov ah, 0dh     ; get pixel.
        int 10h
        xor al, 1111b   ; pixel color
        mov oldX, cx
        mov oldY, dx

draw_pixel:
        mov ah, 0ch     ; set pixel
```

```
        int 10h

check_esc_key:
        mov dl, 255
        mov ah, 6
        int 21h
        cmp al, 27      ; esc?
        jne check_mouse_button
stop:
        ;mov ax, 2  ; hide mouse cursor.
        ;int 33h
        mov ax, 3 ; back to text mode: 80x25
        int 10h
        ; show box-shaped blinking text cursor:
        mov ah, 1
        mov ch, 0
        mov cl, 8
        int 10h
        mov dx, offset msg
        mov ah, 9
        int 21h
        mov ah, 0
        int 16h
        ret
msg db " press any key....     $"
end proses
```

3. Simpan Program yang anda ketik di atas dengan `nama_program.asm`.
4. Gunakan perintah TASM dan TLINK.
5. Jalankan program.

EVALUASI (POST TEST)

Analisis Program di atas dengan menggunakan trace. Jelaskan perubahan yang terjadi!

PRAKTIKUM 07 : PEMROGRAMAN ASSEMBLY MENGGUNAKAN EMU8086

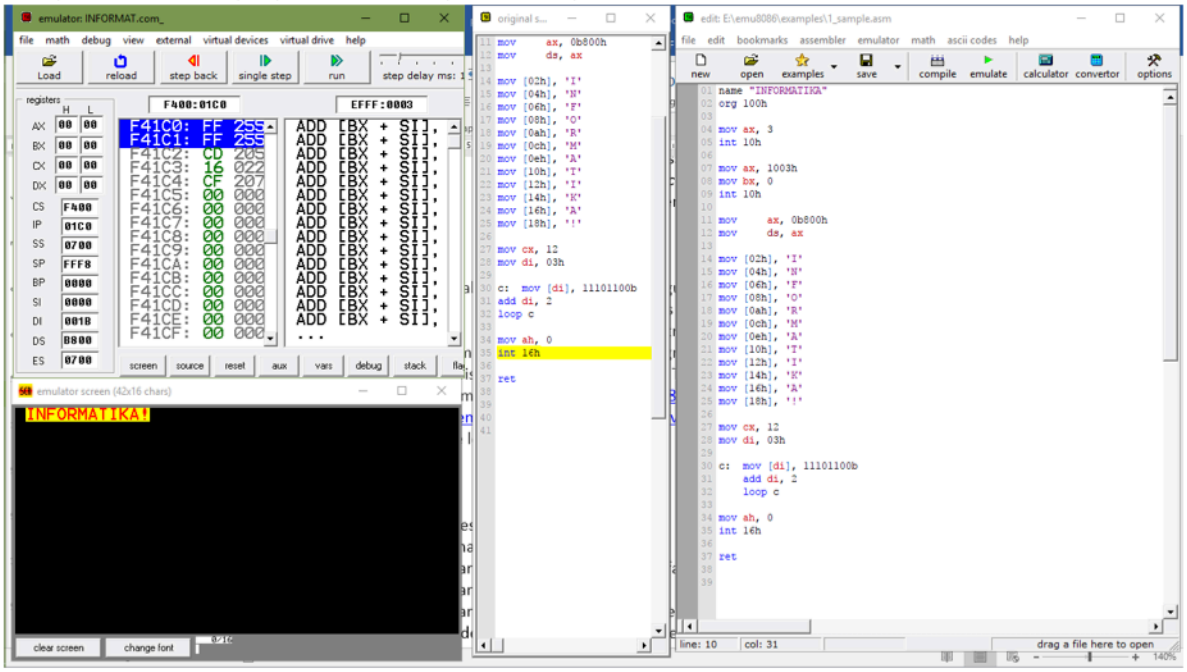
TUJUAN

1. Mahasiswa mampu mengenal salah satu *tools* pemrograman bahasa assembly.
2. Mahasiswa mampu menggunakan editor emu8086 untuk menulis kode bahasa assembly.
3. Mahasiswa mampu meng-*compile* dan menjalan program assembly sederhana dengan emu8086.

DASAR TEORI

Emu8086 merupakan salah satu tools yang dapat digunakan untuk pemrograman assembly. Emu8086 merupakan aplikasi editor sekaligus emulator (debug, simulasi) bagi pemrograman bahasa assembler atau mikrokontroler. Dengan menggunakan aplikasi Emu8086, kita dapat mensimulasikan apakah program yang kita buat sudah benar atau masih salah sekaligus bisa melakukan debugingnya. Tools emu8086 dapat diperoleh secara gratis di web pengembangnya pada halaman http://emu8086.com atau download langsung programnya di http://emu8086.com/files/emu8086v408r11.zip

Tampilan mode lengkap emu8086 dapat dilihat pada Gambar 8.



Gambar 8. Tampilan emu8086

ALAT DAN BAHAN

1. Komputer.
2. Emu8086.

PETUNJUK PRAKTIKUM

1. Buka aplikasi Emu8086.
2. Ketikkan code program berikut di editor emu8086, setelah mengklik tombol **NEW** dan template **Empty Workspace**

```
name "INFORMATIKA"
org 100h
```

```
   mov ax, 3
   int 10h
   mov ax, 1003h
   mov bx, 0
   int 10h
   mov ax, 0b800h
   mov ds, ax
   mov [02h], 'I'
   mov [04h], 'N'
   mov [06h], 'F'
   mov [08h], 'O'
   mov [0ah], 'R'
   mov [0ch], 'M'
   mov [0eh], 'A'
   mov [10h], 'T'
   mov [12h], 'I'
   mov [14h], 'K'
   mov [16h], 'A'
   mov [18h], '!'
   mov cx, 12
   mov di, 03h
c:  mov [di], 11101100b
   add di, 2
   loop c
   mov ah, 0
   int 16h
Ret
```

3. Klik di sistem menu tombol/menu: Emulate, di jendela emulator: klik Run dan amati jendela output.

EVALUASI (POST TEST)

1. Modifikasi prakt7.asm sehingga mampu memunculkan tulisan INFORMATIKA-UAD-JAYA

PRAKTIKUM 08 : PENELUSURAN PROGRAM ASSEMBLY MENGGUNAKAN EMU8086

TUJUAN

1. Mahasiswa mampu mengenal fitur-fitur debuging emu8086
2. Mahasiswa mampu menggunakan emu8086 untuk melakukan tracing/penulusuran kode program assembly dan langsung mengamati luaran yang dihasilkan.

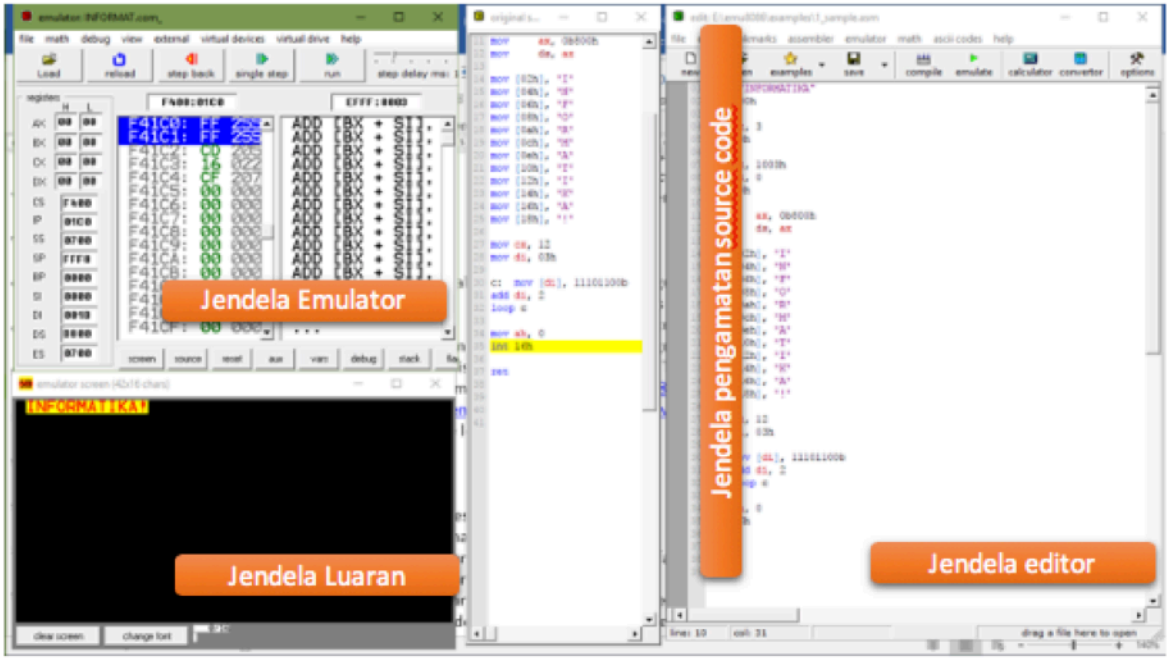
DASAR TEORI

Emu8086 dirancang sebagai emulator yang salah satunya dimanfaatkan sebagai tools untuk melakukan debuging program assembly. Proses tracing/ penelusuran baris demi baris program assembly dapat diamati secara langsung luarannya sesuai dengan perintah pada baris program yang bersangkutan. Keunggulan lainnya di samping dapat menelusuri baris demi baris dan hasil keluarannnya, satu fitur visualisasi proses yang terjadi dalam prosesor (terutama penggunaan alamat register) dapat dipantau juga.

Fitur yang terkait hal tersebut dalam emu8086 disediakan beberapa menu, antara lain:

1. **Compile,** berguna untuk membuat (compile) "binary executable file" atau ekstensi ".com" atau ".exe" dari kode program.
2. **Emulate**, berguna untuk menjalankan emulator kode program yang kita buat
3. **Run**, menjalankan aplikasi dari emulator yang dihasilkan (execute).
4. **Single Step**, berfungsi untuk menjalankan aplikasi dengan cara *tracing* (diproses perbaris kode program).
5. **Step Delay** (dlm milidetik), yang berfungsi untuk memperlambat eksekusi jalannya program (baris program) sehingga bisa dipantau lebih seksama.

Tampilan mode lengkap emu8086 dapat dilihat pada Gambar 9.



Gambar 9. Tampilan emu8086

ALAT DAN BAHAN

1. Komputer.
2. Emu8086.

PETUNJUK PRAKTIKUM

1. Bukalah hasil modifikasi dari praktikum ke-7 (hasil tugas post test praktikum ke-7) dalam editor emu8086 (dengan menu open)
2. Klik di sistem menu tombol/menu: Emulate, akan tampil jendela emulator
3. Lakukan tracing/ penelusuran secara:
    a. Otomatis, dengan tombol Run, angkahnya:
        i. Di Jendela emulator: klik dan geser step delay/ waktu penundaan sesuai selera (dalam mili detik)
        ii. Di jendela emulator: klik Run dan amati jendela output yang ditampilkan.
    b. Manual, dengan tombol **Single Step** dan **Step Back**, langkahnya:
        i. Di jendela emulator: klik **Single Step** dan amati jendela output yang ditampilkan.
        ii. Penelusuran akan berlangsung baris demi baris dan pengamatan luaran bisa lebih seksama
        iii. Lakukan secara berulang klik **Single Step** hingga program berakhir, dan jika dalam jendela luaran meminta interaksi dengan user harus dilakukan tindakan agar proses penelusuran berjalan.
        iv. Jika perlukan untuk kembali ke langkah penelusuran sebelumnya, bisa menggunakan tombol **Step Back**.

EVALUASI (POST TEST)

Lakukan penelusuran dan capture/ screenshot dari beberapa proses di program tersebut (minimal 3 statement program) dan jelaskan analisanya berdasarkan **baris program-alamat register-output program.**

DAFTAR PUSTAKA

1. S'to, 2001, Pemrograman Bahasa Assembly versi 1.0, Jasakom.
2. "January 2003 Laboratory Notes: Computer Engineering II Chapter 10 I/O Devices."
3. Tanenbaum, Andrews, 1991, Computer Network Third Edition, Prentice Hall